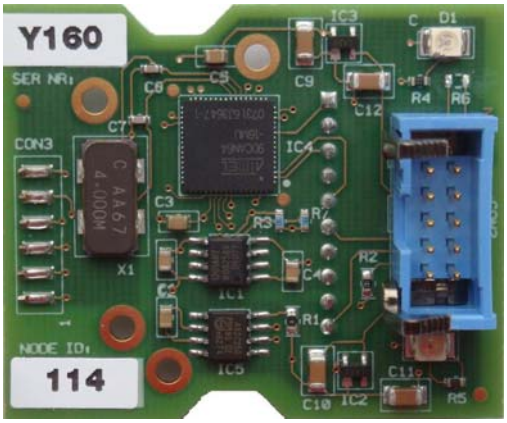
BATCAN

module for single B-sensor read-out by CAN bus and *CANopen*



user manual & reference
v1.3, 5 July 2012

Henk Boterenbrood
NIKHEF, Amsterdam, NL

ABSTRACT

This document describes the BATCAN module, a plug-on for the NIKHEF B-sensor module providing it with a CAN-bus interface. It also describes the firmware features including the CANopen message protocol and CANopen Object Dictionary.

Table of Contents

1 INTRODUCTION AND OVERVIEW 3

2 CONNECTORS AND INTERFACES 4

3 INITIALISATION 7

4 NODE GUARDING AND LIFE GUARDING 8

5 B-SENSOR DATA READ-OUT 10

5.1 INTRODUCTION 10
5.2 B-SENSOR DATA 11
5.3 ADC DATA CONVERSION 13
5.4 B-SENSOR SERIAL NUMBER 14

6 CONFIGURATION STORAGE 16

6.1 STORING PARAMETERS AND SETTINGS 16
6.2 EEPROM MEMORY MAP 17

7 UPGRADING THE FIRMWARE 18

8 BATCAN OBJECT DICTIONARY 19

9 EMERGENCY OBJECTS 26

REFERENCES 28

APPENDIX A. NTC TEMPERATURE SENSOR DATA 29

Version History		
Version	Date	Comments
1.3	5 Jul 2012	Describes firmware version "BC11.0002". Added Object 5380h and 2500h, sub 29-37.
1.2a	9 Feb 2011	Fixed some errors in schematic in Figure 1.
1.2	4 Feb 2010	Describes firmware version "BC11.0000". Additional Objects for compatibility with *BATsCAN* firmware for *mBATCAN* modules. Added Object 2800h to OD listing, was missing. Object 2910h renamed to 29F0h.
1.1	26 Sep 2008	Describes firmware version "BC10.0003". Additional Objects 2900h, sub 3 and 2910h. New pictures; some changes to text.
1.0	14 Aug 2008	Describes firmware version "BC10.0000".

Table 1. Document change record (latest change first).

1 Introduction and Overview

The ***BATCAN*** module is a module that plugs directly on the NIKHEF **B-sensor module** [1], providing a CAN interface for reading out the B-sensor module. The BATCAN firmware is based on the firmware used in the ATLAS MDT-DCS module [2], and is –as far as the read-out of a single B-sensor module is concerned– completely compatible with that module.

In addition the BATCAN module may be used as a component in the so-called ***BsCAN3*** system [3], in which a number of ***BATCAN*** and ***mBATCAN*** [4] modules make up a B-field sensor measurement system with multiple B-sensor modules, in closely spaced groups read out by an mBATCAN module and/or individual modules typically spaced further apart, each read out by a BATCAN module, all connecting to a single CAN bus which can extend to over 100 m in length.

Some additions have been made to the firmware in order to make the BATCAN module compatible with the mBATCAN module from the CANopen point-of-view.

A schematic of the BATCAN module is shown in Figure 1. The central component of the module is an Atmel AT90CAN64 microcontroller with integrated CAN-controller (hence the name **BATCAN**: ***B**-sensor readout module with **AT**mel **CAN** microcontroller*)

The BATCAN module controls and monitors the B-sensor module ADC and provides read-out of the B-sensor data via a CAN bus. The ***CANopen*** protocol [5] [6] is used as high-level communication protocol standard on the bus.

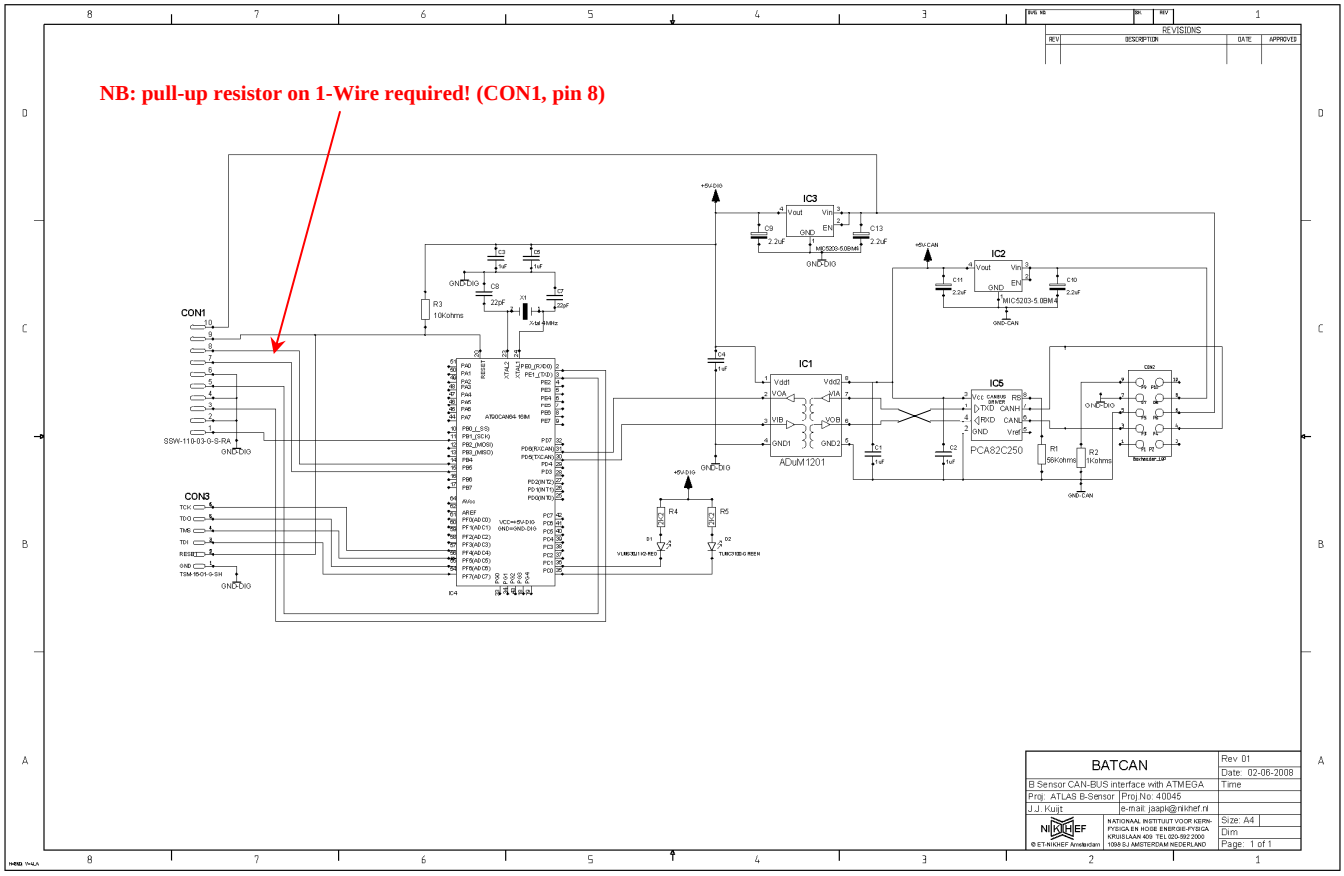


Figure 1. Schematic of the BATCAN module (NB: modified original in driver/coupler area).

2 Connectors and Interfaces

The pictures in Figure 2 show the BATCAN module and its external interfaces, and plugged onto a B-sensor module.

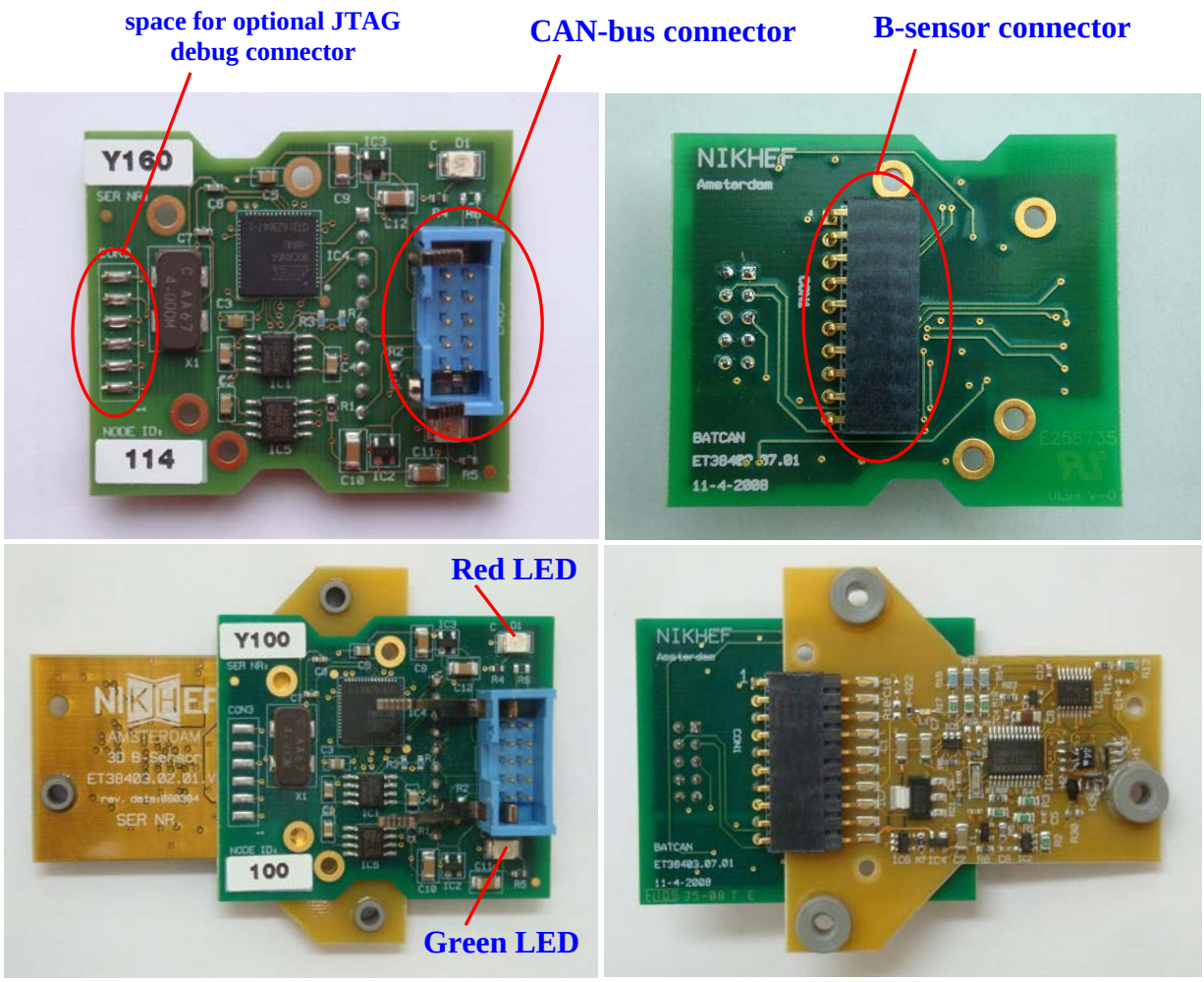
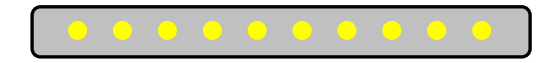


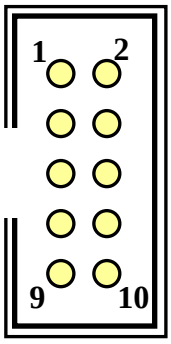
Figure 2. The BATCAN module (top left & right), and plugged onto a B-sensor module (bottom left & right).



Pin	B-sensor function	Comment
1	***SCLK***	SPI Serial Clock (to ADC)
2	GND	
3	***SDI***	SPI Serial Data In (to ADC)
4	GND	
5	***SDO***	SPI Serial Data Out (from ADC)
6	GND	
7	***CS***	Chip Select (to ADC)
8	***ID***	1-Wire interface (to ID-chip)
9	–	
10	V+	from CAN-connector (pin 8)

Pin	Program function	Comment
1	***SCK***	Programming Clock
2	GND	
3	***PDI***	Program Data In
4	GND	
5	***PDO***	Program Data Out
6	GND	
7	–	
8	–	
9	***RESET***	Reset to the AT90CAN micro
10	V+	from CAN-connector (pin 8)

Table 2. Layout of the *B-sensor/Programming* connector. The programming connections are used only once, to program the Bootloader in the microcontroller's flash memory; from then on the CAN bus is used to upload application firmware. (Note that e.g. the AVRISP programmer needs power within the range 2.7-5.5V, so do not connect it directly to V+, which normally lies in the range 6-12V).



function	pin	pin	function
not connected	**1**	**2**	*not connected* (***CAN-GND***)
CAN-L	**3**	**4**	**CAN-H**
CAN-GND	**5**	**6**	**+V** (6-12V)
GND	**7**	**8**	**CAN-POWER** (6-12V)
CAN-SHIELD	**9**	**10**	*not connected*

Table 3. Layout of the BATCAN *CAN* connector. *CAN-POWER* powers the CAN-driver part of the BATCAN module. *+V* powers the rest of the BATCAN module as well as the B-sensor module.

The BATCAN module features two LEDs to indicate the status of the module, a red one and a green one (see Figure 2). The green LED is on when the module has properly initialized and the firmware is running; it blinks briefly when CAN-messages are received or sent. The red LED comes on when there is a problem with (reading out) the B-sensor.

The BATCAN module's serial number, which it has been given after production testing, can be read out remotely (see *Object Dictionary* index 3100h in section 8).

The module's CAN node identifier is stored in EEPROM and can be changed remotely (see *Object Dictionary* index 3300h and 3301h in section 8).

3 Initialisation

When the BATCAN firmware starts up, the hardware devices are reset and configured (i.e. the CAN-controller and the ADC on the B-sensor module) and error counters and registers are reset.

After power-up, watchdog reset, manual reset or a *CANopen* initiated reset action (i.e. by an NMT *Reset-Node* message, see below) a CANopen node sends a so-called ***Boot-up*** message (as defined by the *CANopen* standard) as soon as it has finished initializing (hardware, software); this is a CAN-message with the following syntax:

BATCAN module (NMT-Slave) → Host (NMT-Master)

COB-ID	Data Byte 0
700h + *NodeID*	0

NodeID is the CAN node identifier stored in the BATCAN's EEPROM. *NodeID* is in the range between 1 and 127.

To *start* the BATCAN application in the CANopen sense of the word, the following CANopen NMT message must be sent:

Host (NMT-Master) → BATCAN module (NMT-Slave)

COB-ID	Data Byte 0	Data Byte 1
000h	01h (*Start_Remote_Node*)	*NodeID* or 0 (0: all nodes on the bus)

There is no reply to this message.

Now the BATCAN module is *Operational*, meaning that it monitors I/O channels (depending on configuration) and can send and receive (and processes) CANopen ***PDO*** messages, which carry the application data (see next sections).

Optionally a feature called *auto-start* may be enabled, so that the BATCAN module automatically goes to *Operational* state after power-up or reset. The *auto-start* feature can be configured in *Object Dictionary* index 3200h, subindex 2.

To generate a *soft* reset the following *CANopen* NMT message must be sent:

Host (NMT-Master) → BATCAN module (NMT-Slave)

COB-ID	Data Byte 0	Data Byte 1
000h	81h (*Reset_Node*)	*NodeID* or 0 (0: all nodes on the bus)

Again, there is no reply to this message.

Note that at power-up it is the *Bootloader* application firmware that becomes active first and is in control of the BATCAN module; the Bootloader reports its presence by sending the following Emergency message (see also section 7):

Bootloader → Host

COB-ID	Byte 0-1	Byte 2	Byte 3-7
080h + *NodeID*	Emergency Error Code (00h 50h)	Error Register (Object 1001h) (80h)	Manufacturer specific error field (FEh 00h 64h ZZh 00h) (ZZh = MCUSR)

(*MCUSR* = MCU Status Register; for details see section 9 or the AT90CANxx datasheet).

Having the Bootloader activate at power-up guarantees that it is always possible to upload new application software to the module, even when the application currently programmed is faulty or corrupted.

After about 4 seconds the Bootloader automatically jumps to the application. Alternatively, the Bootloader starts the application immediately, if it receives an NMT *Reset-Node* message –as shown above- within this period.

4 Node Guarding and Life Guarding

Node Guarding in CANopen is a mechanism whereby an *NMT-master* checks the state of other nodes on the bus, at regular intervals. It can do this in one of two different ways:

1. The master sends a Remote Transmission Request (RTR) for the Node Guard message, to each node on the bus, in turn; a node that receives the RTR, sends the Node Guard message, which contains one data byte indicating the (CANopen) state of the node, as well as a toggle bit. If a node does not reply the master should signal this to the higher-level software and/or take appropriate action.
   The RTR for the Node Guard message looks like this (a Remote Frame, so the CAN-message has no data bytes):

   Host (NMT-Master) → BATCAN module (NMT-Slave)

COB-ID
700h + *NodeID*

   The reply Node Guard message from a node looks like this:

   BATCAN module (NMT-Slave) → Host (NMT-Master)

COB-ID	DataByte 0
700h + *NodeID*	bit 7: *toggle bit*, bit 6-0: *state*

2. Each node on the bus sends a Heartbeat message at regular intervals; typically, the NMT-master monitors these messages and keeps a time-out period for each node. The master detects nodes that stop sending their Heartbeat messages and should signal this to the higher-level software and/or take appropriate action.
   A Heartbeat message looks like this:

BATCAN module (Heartbeat producer) → Consumer(s) (e.g. NMT-Master)

COB-ID	DataByte 0
700h + *NodeID*	*State*

State is one of these CANopen states: 0 (*Initializing*), 4 (*Stopped*), 5 (*Operational*) or 127 (*Pre-operational*). Note that this makes the *Boot-up* message the first Heartbeat message after a node reset (see previous section).

According to the CANopen standard, a node is not allowed to support both Node Guarding and Heartbeat protocols at the same time. The BATCAN module supports both methods of Node Guarding (but indeed not at the same time), i.e. it can send the Node Guard message or it can send the Heartbeat message with an interval, which is configurable in *Object Dictionary* index 1017h.

Life Guarding in CANopen is a mechanism whereby a node checks the aliveness of the host or master, by applying a time-out on messages received. CANopen defines that the message to time-out is the RTR for the Node Guard message, sent by the NMT-master; however, the BATCAN module resets its Life Guarding timer at each properly received message addressed to it.

Life Guarding is controlled through *Object Dictionary* objects 100Ch and 100Dh. In the BATCAN module the Life Guarding time-out can be set between 1 and 255 seconds, by setting *Object Dictionary* index 100Dh to the corresponding value, or can be switched off, by setting *Object Dictionary* index 100Dh to zero.

If a Life Guarding time-out occurs, the node should take whatever appropriate action. The BATCAN module resets and reinitializes the CAN-controller, and (tries to) resume(s) normal operation, after sending an Emergency message (see section 9).

5 B-sensor Data Read-out

5.1 Introduction

Each data object in the BATCAN module can be accessed through the CANopen *Object Dictionary* (*OD*). The CANopen ***SDO*** (Service Data Object) confirmed message mechanism is used to read from and write to data objects in the *OD*.
A complete overview of the *Object Dictionary* of the BATCAN module can be found in section 8.

A more efficient method of read-out of data from the BATCAN module is offered by the CANopen mechanism of ***PDO*** (Process Data Object) messages. This is an unconfirmed message mechanism without protocol overhead, and thus much more suitable for regular monitoring of the *process* data of the BATCAN module, i.e. the B-sensor data. The sending of this type of message may be triggered by a host system or autonomously by the BATCAN module firmware.

From the point of view of the BATCAN module data are transmitted by a ***PDO*** message, called a ***Transmit-PDO*** (or ***TPDO***). In CANopen the CAN-identifier, message content and *transmission type* of PDO messages may be configurable (configure by writing to the appropriate objects in the Object Dictionary using the SDO mechanism).
However, the CANopen standard defines a predefined set of CAN-identifiers (the so-called ***Predefined Connection Set***), defining which CAN-identifier to use for which kind of CANopen message, without the need for the node to support configuration. The BATCAN module uses this set of identifiers. Also the PDO message content is fixed in the BATCAN module and cannot be changed. The content of PDO messages can be found and read from the *OD* from objects called *PDO mapping objects* (stored at fixed entries in the *OD*).
A feature that *is* configurable on the BATCAN module is the so-called *transmission type* of the TPDOs, which controls what triggers it to send its 'process' data, e.g. periodically, on request or on-change. This is described in a next section.

Serious problems occurring during read-out, e.g. with the ADC hardware, are reported in so-called CANopen ***Emergency*** messages. A list of the *Emergency* messages the BATCAN module can produce is given in section 9, including a description of the problem.

5.2 B-sensor Data

The BATCAN module sends one PDO message containing 5 bytes for each B-sensor input and four of the B-sensor's inputs are read: Hall sensors H1, H2 and H3 and the temperature sensor. The CAN-identifier used for this PDO is the so-called *4th-transmit-PDO* (**TPDO4**) of the CANopen *Predefined Connection Set*.

The BATCAN module produces the following 5-databyte TPDO4 [1]:

BATCAN module → Host

COB-ID	Data Byte 0	Data Byte 1	Data Byte 2-4
480h + *NodeID*	Channel number	ADC-config	24-bit ADC value

with:

ADC value: Signed/unsigned 24-bits ADC value, LSB in byte 2, MSB in byte 4.
Note: Hall sensors: 24-bit signed value; T-sensor: 24-bit unsigned value (either an ADC count or a temperature in millidegrees centigrade depending on the setting of *OD* index 4400h),

Channel number: Number between 0 and 3.
Chan 0-3: Hall sensor H1, H2, H3 and T-sensor resp. of the B-sensor

ADC-config: **bit 7**: not used.
bits 6-0: ADC configuration: conversion word rate (bits W0, W1 and W2), gain range (bits G0, G1 and G2) and unipolar or bipolar (bit U/B); see below. For definitions see *OD* index 2500h/2501h, sub 2,3,4,5,6 and 7.

BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Meaning	-	W2	W1	W0	G2	G1	G0	U/B

The method by which the B-sensor module inputs are read out depends on the *transmission-type* of TPDO4, which can be set in *OD* index 1803h, subindex 2 of the BATCAN module.

The following modes of TPDO2 transmission are supported (see *OD* index 1803h, subindex 2 and 5):

- **PDO transmission type 1**:
  after every so-called **SYNC** message issued on the CAN-bus the BATCAN module starts a B-sensor input channel scan and sends four TPDO4 messages, containing the Hall-sensors and T-sensor data.
  The SYNC message is a CAN-message with a fixed COB-ID and no data bytes:

  Host → all (SYNC-)slave nodes

COB-ID
080h

[1] NB: optionally a 6-byte TPDO4 can be configured in *OD* index 4500h, see section 8; a data byte containing 'index' value 0 is inserted on data byte position 0 of the PDO; this makes the BATCAN's PDO message compatible with PDO messages with B-sensor data from nodes with ***BATsCAN*** firmware used in ***mBATCAN*** nodes, see [4].

Note that all nodes that have PDOs configured to respond to a SYNC message will respond to the SYNC, which is a broadcast message.

- **PDO transmission type 255**:
  after every so-called Remote Transmission Request (**RTR**) for TPDO4 the BATCAN module starts a B-sensor input channel scan and sends four TPDO4 messages, containing the Hall-sensors and T-sensor data. The *Remote Frame* CAN-message that constitutes this RTR has no data bytes and looks like this:

  Host → BATCAN module

COB-ID
280h+*NodeID*

  Note that an RTR is sent to and received/processed by only one particular node.

- **Event Timer > 0**:
  If TPDO4's *event timer* (*OD* index 1803h, sub 5) is set to a value unequal to zero (*event timer* is expressed in units of 1 s and must be <=255) the BATCAN module automatically starts a B-sensor input channel scan periodically, triggered by a timer (in this mode an RTR or SYNC message also triggers an input scan, depending on the transmission mode as shown above).

Optionally a reset and calibration sequence can be done before each B-sensor ADC channel scan. This feature can be enabled via *OD* index 2700h (useful perhaps for increasing radiation tolerance).

Individual B-sensor module channels (there are actually 7 inputs) can be read out using CANopen **SDO** messages by reading from *OD* index 4200h (see *OD* tables for a description of each individual channel).

5.3 ADC Data Conversion

The interpretation of the Hall sensor ADC values and conversion to physical values will be done offline using a set of calibration tables for each individual B-sensor module and some dedicated software (available from Felix Bergsma at CERN).

The B-sensor module's T-sensor is an NTC, *Thermometrics* type number DC95F502W, with a nominal resistance of 5 kΩ. See Appendix A for datasheet and temperature data of the NTC.

Table 4 shows a list of resistance values $\mathbf{R_{NTC}}$ for this NTC at different temperatures, and the resulting B-sensor module ADC input voltage. In the shaded part of the table (between 0º and 70º C) the precision is ±0.2º C.

The ADC input voltage $\mathbf{V_{NTC}}$ can be expressed as:

$$\mathbf{V_{NTC} = V_{ref} - V_{cc}R_{NTC} / (R_{NTC} + R_{ref})}$$

which can be rewritten as:

$$\mathbf{R_{NTC} = R_{ref} (V_{ref} - V_{NTC}) / (V_{NTC} + Vcc - V_{ref})}$$

With $\mathbf{R_{ref}}$ = 23.2 kΩ, $\mathbf{V_{cc}}$ = 5 V and $\mathbf{V_{ref}}$ = 2.5 V this results in:

$$\mathbf{R_{NTC} = 23200 (2.5 - V_{NTC}) / (V_{NTC} + 2.5)}$$

$\mathbf{V_{NTC}}$ is the voltage value calculated from the 24-bit ADC value **A.**
The ADC input has been calibrated to give **A**=0 (000000h) at 0 ºC (i.e. at 0.4315 V) and **A**=16777215 (0xFFFFFF) at 100 ºC (i.e. at 2.4275 V), so that $\mathbf{V_{NTC}}$ can be expressed as:

$$\mathbf{V_{NTC} = 0.4315 + (2.4275 - 0.4315)A/FFFFFFh = 0.4315 + 1.996A/FFFFFFh}$$

So $\mathbf{R_{NTC}}$ can be calculated directly from ADC value **A** as follows:

$$\mathbf{R_{NTC} = 23200 (2.0685 - a) / (2.9315 + a)}$$

with **a** = 1.996 **A** / 16777215.

To calculate temperature **T** (in ºC, in the range from 0 to 100 ºC) of the NTC from NTC resistance value $\mathbf{R_{NTC}}$ (in Ω), the following approximation equation (see Appendix A) is used:

$$\mathbf{T = (1.0 / (a + b \ln(r) + c (\ln(r))^2 + d (\ln(r))^3)) - 273.15}$$

with **r** = $\mathbf{R_{NTC}}$/5000,
and **a** = 3.3538646E-03
b = 2.5654090E-04
c = 1.9243889E-06
d = 1.0969244E-07

when 68.600 >= **r** > 3.274 (i.e. when -50º C <= **T** < 0º C),
or **a** = 3.3540154E-03
b = 2.5627725E-04
c = 2.0829210E-06
d = 7.3003206E-08
when 3.274 >= **r** > 0.36036 (i.e. when 0º C <= **T** < 50º C),
or **a** = 3.3539264E-03
b = 2.5609446E-04
c = 1.9621987E-06
d = 4.6045930E-08
when 0.36036 >= **r** >= 0.06831 (i.e. when 50º C <= **T** < 100º C).
or **a** = 3.3368620E-03
b = 2.4057263E-04
c = -2.6687093E-06
d = -4.0719355E-07
when 0.06831 >= **r** >= 0.01872 (i.e. when 100º C <= **T** < 150º C).

The conversion functions above are applied by the BATCAN firmware to the ADC readings when temperature read-out is set to millidegrees centigrade, which is the default.

5.4 B-sensor Identification Number

Each B-sensor module comes equipped with a unique identification number, which is factory-lasered in the on-board Dallas DS2405 device.
The 64-bit (8-byte) number is used to uniquely identify each module, for instance to match each module with its calibration data, which are kept elsewhere off-line.

The identification number of the B-sensor modules can be read from *OD* Object 2900h. The least significant 4 bytes are read from subindex 1 and the most significant 4 bytes from subindex 2. The least- or most-significant sets of 4 bytes can be read in any order.

The layout of the 64-bit ID number is as shown below:

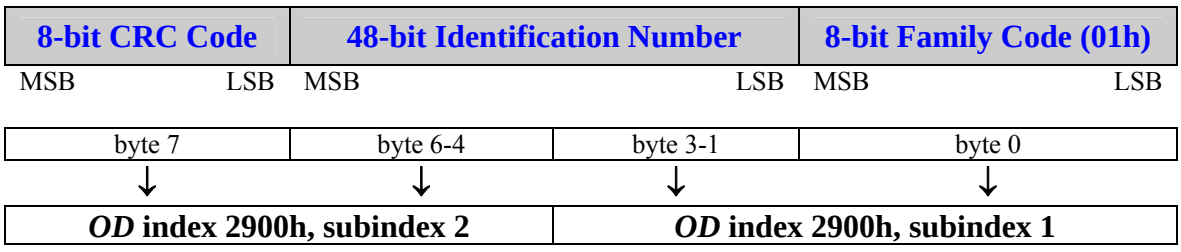


Figure 3. B-sensor 64-bit ID Number and its mapping to Object Dictionary (*OD*) objects.

The BATCAN module checks the correctness of the identification number CRC when *OD* Object 2900h is read, so a valid reply implies the CRC was correct: the host does not need to recalculate the number's CRC.

Temperature	Normalized Resistance	Resistance	AIN4 (ADC)
[C]	Ohm	Ohm	Volt
-50	68.60	343000.00	-2.1832
-45	48.16	240800.00	-2.0606
-40	34.23	171150.00	-1.9031
-35	24.62	123100.00	-1.7071
-30	17.91	89550.00	-1.4712
-25	13.17	65850.00	-1.1974
-20	9.782	48910.00	-0.8913
-15	7.339	36695.00	-0.5633
-10	5.558	27790.00	-0.2250
-5	4.247	21235.00	0.1106
0	3.274	16370.00	0.4315
5	2.544	12720.00	0.7294
10	1.992	9960.00	0.9982
15	1.572	7860.00	1.2347
20	1.250	6250.00	1.4389
25	1.000	5000.00	1.6135
30	0.8056	4028.00	1.7603
35	0.6530	3265.00	1.8831
40	0.5326	2663.00	1.9852
45	0.4369	2184.50	2.0697
50	0.3604	1802.00	2.1396
55	0.2989	1494.50	2.1974
60	0.2491	1245.50	2.2452
65	0.2087	1043.50	2.2848
70	0.1756	878.00	2.3177
75	0.1485	742.50	2.3449
80	0.1261	630.50	2.3677
85	0.1075	537.50	2.3868
90	0.09209	460.45	2.4027
95	0.07916	395.80	2.4161
100	0.06831	341.55	2.4275
105	0.05916	295.80	2.4371

Table 4. NTC resistance/temperature table, and resulting B-sensor ADC input voltage (Normalized resistance table taken from the datasheet, see Appendix A).

6 Configuration Storage

6.1 Storing Parameters and Settings

Parameters and settings can be stored permanently onboard in non-volatile memory (EEPROM) by writing string "save" to *OD* index 1010h. The ***SDO*** mechanism is used to accomplish this, using the following message:

Host → BATCAN module

COB-ID	**Data Byte**							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
600h + *NodeID*	0x23	0x10	0x10	*subindex*	73h ('s')	61h ('a')	76h ('v')	65h ('e')

with *OD* index 1010h in byte 1+2 and *subindex* in byte 3 with *subindex*:
= 1: store all parameters (as listed for *subindex* 2 and 3).
= 2: store communication parameters (concerning CAN, PDOs and Node- and Life Guarding).
= 3: store application parameters (concerning ADCs, Digital I/O and JTAG).
= 4: see next section.

If the store-operation succeeded the BATCAN module sends the following reply:

BATCAN module → Host

COB-ID	**Data Byte**						
	0	**1**	**2**	**3**	**4**	**5**	**6-7**
580h + *NodeID*	0x60	0x10	0x10	*subindex*	–	–	–

If the store-operation did <u>*not*</u> succeed the BATCAN module sends the following reply (***SDO** Abort Domain Transfer*, error reason: 'hardware fault' (for more details see [5])):

BATCAN module → Host

COB-ID	**Data Byte**							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
580h + *NodeID*	80h	10h	10h	*Subindex*	0	0	6 (Error Code)	6 (Error Class)

Parameters can be reset to their default values (by invalidating the corresponding contents of the EEPROM) by writing to *OD* index 1011h, using this time the string "load" (6Ch, 6Fh, 61h, 64h) in bytes 4 to 7 of the ***SDO***. Note that the default values take effect only after a subsequent reset of the module. The default parameter values are listed in the *OD* tables in section 8.

The *Object Dictionary* tables in section 8 show which settings can be stored in EEPROM: these are marked by an asterisk (*) in the first column

(Note that storage of the BATCAN Serial Number is handled separately).

6.2 EEPROM Memory Map

Table 5 below details the layout of the AT90CAN64 microcontroller EEPROM usage by the BATCAN application firmware.

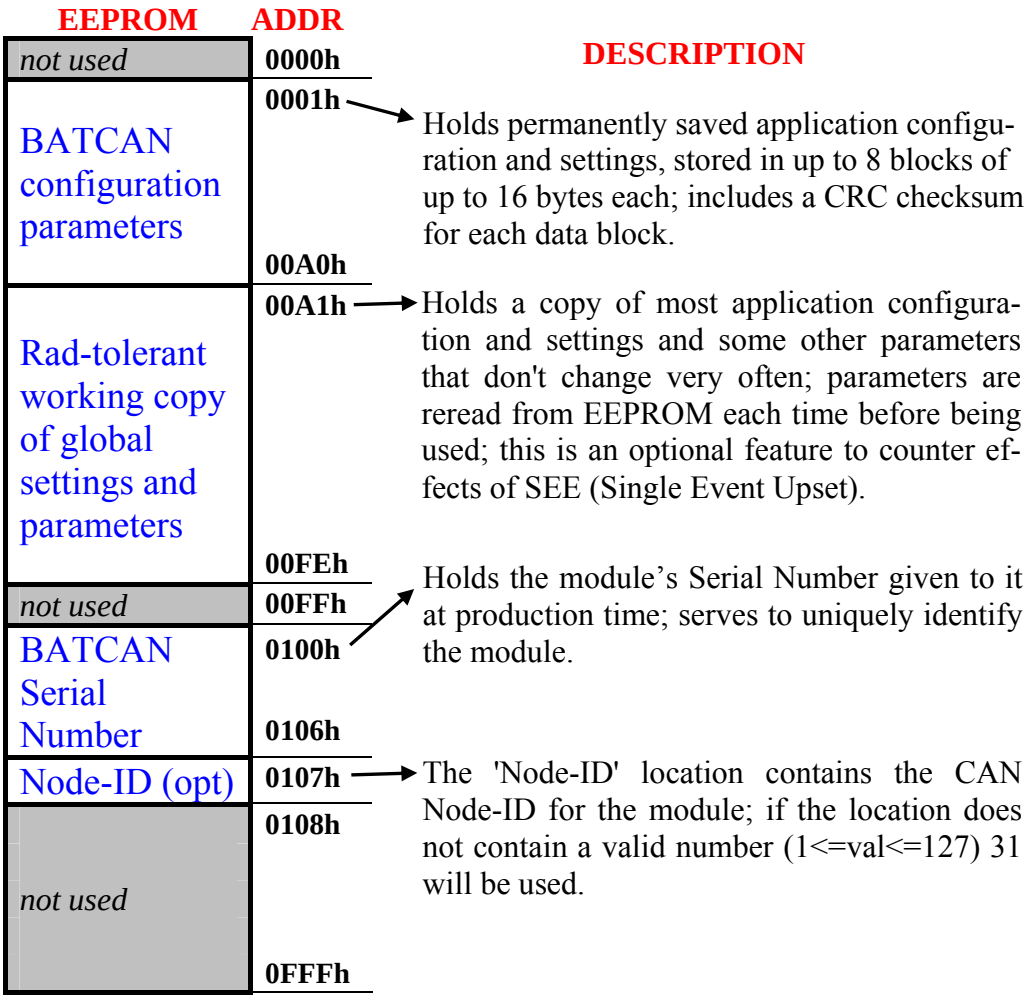

EEPROM
ADDR
DESCRIPTION
not used
0000h
0001h
BATCAN configuration parameters
Holds permanently saved application configuration and settings, stored in up to 8 blocks of up to 16 bytes each; includes a CRC checksum for each data block.
00A0h
00A1h
Rad-tolerant working copy of global settings and parameters
Holds a copy of most application configuration and settings and some other parameters that don't change very often; parameters are reread from EEPROM each time before being used; this is an optional feature to counter effects of SEE (Single Event Upset).
00FEh
not used
00FFh
0100h
BATCAN Serial Number
Holds the module's Serial Number given to it at production time; serves to uniquely identify the module.
0106h
Node-ID (opt)
0107h
The 'Node-ID' location contains the CAN Node-ID for the module; if the location does not contain a valid number (1<=val<=127) 31 will be used.
0108h
not used
0FFFh

Table 5. AT90CAN64 microcontroller EEPROM memory map of the BATCAN application firmware.

7 Upgrading the Firmware

The application program in the BATCAN microcontroller can be replaced or upgraded by uploading new program code via the CAN-bus.

A Windows application program called **ELMBloader** is available for performing this firmware upgrade. The upgrade process leaves the EEPROM intact, in other words: all existing configuration settings are preserved during an upgrade.

The ***Bootloader*** [7] is an application program stored in a separate section of the microcontroller flash memory. It handles the firmware upgrade process, receiving series of CAN(open) messages containing the programming instructions and code.

After power-up of the BATCAN module, it is the *Bootloader*, that takes control of the module initially. After about 4 seconds the *Bootloader* automatically jumps to the start of the BATCAN application program, or immediately after it receives a CANopen ***NMT** Reset-Node* message. However, the *Bootloader* remains in control if it receives a valid programming command within those 4 seconds. The firmware upgrade process may then begin.

The BATCAN application program can transfer control of the module explicitly to the *Bootloader*, when one writes any value to the 8-bit object 5E00h in the Object Dictionary of the BATCAN application. In this case the *Bootloader* does *not* automatically jump to the BATCAN application program after 4 seconds. The firmware upgrade process may now begin.

After the upgrade process, the reception of a CANopen ***NMT** Reset-Node* message causes the *Bootloader* to jump to the start of the new application program.

If the BATCAN module sends an *Emergency* message as shown below, it signifies that the *Bootloader* is in control of the module. Note that the same Emergency message is also sent as the first message after power-up, when the *Bootloader* is in control for the first 4 seconds after power-up, before jumping to the application program.

The *Bootloader* can be forced to jump to the application immediately, by sending it a CANopen ***NMT*** Reset-Node message.

COB-ID	Byte 0-1	Byte 2	Byte 3-7
080h + *NodeID*	Emergency Error Code (00h 50h)	Error Register (Object 1001h) (80h)	Manufacturer specific error field (5 bytes: FEh,80h,64h,*ZZh*,00h, with *ZZh* = MCUSR)

(*MCUSR* = MCU Status Register contents; for details see section 9).

8 BATCAN Object Dictionary

The values of objects marked with ∗ in the *Index* column can be stored permanently in EEPROM. They are retrieved from EEPROM at reset and power-up.

Communication Profile Area (*BATCAN*)						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
1000	-	Device type	U32	RO	00000000h	Meaning: no specific device profile
1001	-	Error register	U8	RO	0	
1002	-	Manufacturer status reg	U32	RO	0	[1] (see footnote)
1008	-	Manufacturer device name	VisStr	RO	"BATC"	= BATCAN_module
1009	-	Manufacturer hw version	VisStr	RO	"bc10"	= BATCAN v1
100A	0	Manufacturer software version	VisStr	RO	"BC11"	***BATCAN*** application v1.1.2
	1	minor version number	VisStr	RO	"0002"	
100C	-	Guard time [ms]	U16	RO	1000	= 1 second
100D *	-	Life time factor	U8	RW	0	Life Guarding timeout in seconds; 0 → no life guarding timeout
1010		Store parameters	Array			Save stuff in onboard EEPROM
	0	Highest index supported	U8	RO	3	
	1	Save all parameters	U32	RW	1	Read: 1; Write "save": store all
	2	Save communication parameters	U32	RW	1	Read: 1; Write "save": store PDO par's, Life time factor, …
	3	Save application par's	U32	RW	1	Read: 1; Write "save": store ADCs config, …
1011		Restore default parameters	Array			Invalidate stuff in onboard EEPROM; use defaults
	0	Highest index supported	U8	RO	3	
	1	Restore all parameters	U32	RW	1	Read: 1; Write "load": invalidate all parameters stored
	2	Restore communication parameters	U32	RW	1	Read: 1; Write "load": invalidate stored PDO par's, etc.
	3	Restore application par's	U32	RW	1	Read: 1; Write "load": invalidate stored ADCs config, etc.
1017 *	-	Producer Heartbeat Time [1 s]	U16	RW	0	In units of seconds (but <=255 !), (NB: actually should be in ms according to CANopen!); 0 → Heartbeat is disabled
1018		Identity	Record			Mandatory CANopen object
	0	Number of entries	1..4	RO	1	
	1	Vendor ID	U32	RO	12345678h	to be ordered from CiA

[1] Manufacturer Status Register: byte1 = **B-sensor ADC**. (byte1 for compatibility with MDT-DCS app)
Status byte/nibble: **01**: ADC reset error, **02**: ADC calibration error, **04**: ADC conversion time-out, **FF**: ADC absent / not used.

Communication Profile Area (*BATCAN*) *(continued…)*						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
1803		4th Transmit PDO par's	Record			Data type = PDOCommPar
	0	Number of entries	U8	RO	5	
	1	COB-ID used by PDO	U32	RO	480h + *NodeID*	According to CANopen Predefined Connection Set
*	2	Transmission type	U8	RW	1	Only 1 and 255 allowed
	3	Inhibit time [100 µs]	U16	RO	0	*not used*
	4	*Not used*	U8	RO	0	
*	5	Event timer [1 s]	U16	RW	0	In units of secs, must be <= 255; active for all transmission-types!
1A03		4th Transmit PDO mapping	Record			Data type = PDOMapping
	0	Number of entries	U8	RO	2	*should be 255 for MuxPDO, but this is not a CANopen MPDO…*
	1	B-sensor ADC channel number	U32	RO	42000008h	*actually not allowed, but…*
	2	24-bit analogue input	U32	RO	420*x*0*x*20h	OD-index 4200/4201,subindex *x*, Analogue inputs, multiplexed, size = 32 bits

Manufacturer-specific Profile Area (*BATCAN*) *(continued…)*						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
2500		B-sensor ADC-config	Record			CS5524 24-bit ADC
	0	Number of entries	U8	RO	22	
	1	Number of input channels	U8	RO	7	
*	2	Conversion Word Rate Hall	U8	RW	0	3-bit code [1]
*	3	Input Voltage Range Hall	U8	RW	0	3-bit code [2]
*	4	Unipolar/Bipolar Measurement Mode Hall	U8	RW	0	0 = bipolar, 1 = unipolar
*	5	Conversion Word Rate Temp	U8	RW	0	3-bit code [1]
*	6	Input Voltage Range Temp	U8	RW	5	3-bit code [2]
*	7	Unipolar/Bipolar Measurement Mode Temp	U8	RW	1	0 = bipolar, 1 = unipolar
	8	Power Save Mode	Bool	WO		1 = set ADC to power save mode 0 = take ADC out of this mode
	9/29[3]	Configuration Register	U32	RW		CS5524 Config Register
	10/30[3]	Offset Register #1	U32	RW		CS5524 physical channel AIN1
	11/31[3]	Gain Register #1	U32	RW		CS5524 physical channel AIN1
	12/32[3]	Offset Register #2	U32	RW		CS5524 physical channel AIN2
	13/33[3]	Gain Register #2	U32	RW		CS5524 physical channel AIN2
	14/34[3]	Offset Register #3	U32	RW		CS5524 physical channel AIN3
	15/35[3]	Gain Register #3	U32	RW		CS5524 physical channel AIN3
	16/36[3]	Offset Register #4	U32	RW		CS5524 physical channel AIN4
	17/37[3]	Gain Register #4	U32	RW		CS5524 physical channel AIN4
	18	Channel-Setup Register #1	U32	RW		LC 1 (12-bits) in lower 2 bytes, LC 2 (12-bits) in upper 2 bytes
	19	Channel-Setup Register #2	U32	RW		LC 3 (12-bits) in lower 2 bytes, LC 4 (12-bits) in upper 2 bytes
	20	Channel-Setup Register #3	U32	RW		LC 5 (12-bits) in lower 2 bytes, LC 6 (12-bits) in upper 2 bytes
	21	Channel-Setup Register #4	U32	RW		LC 7 (12-bits) in lower 2 bytes, LC 8 (12-bits) in upper 2 bytes
*	22	SPI SCLK signal high period (opto-coupler delay)	U8	RW	10	in µs, 10 <= value <= 255
	23-28	---	---	---		reserved

[1] **000**: 15.0 Hz, **001**: 30.0 Hz, **010**: 61.6 Hz, **011**: 84.5 Hz,
100: 101.1 Hz, **101**: 1.88Hz, **110**: 3.76 Hz, **111**: 7.51 Hz

[2] **000**: 100 mV, **001**: 55 mV, **010**: 25 mV, **011**: 1 V, **100**: 5 V, **101**: 2.5 V

[3] Entries for *BATsCAN* firmware compatibility.

Manufacturer-specific Profile Area (*BATCAN*) *(continued…)*						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
2600	-	ADC-reset-and-calibrate B-sensor	U8	WO		Writing any value triggers a reset and calibration sequence on the B-sensor with its current ADC settings
2700 *	-	ADC-reset-and-calibrate before every scan cycle	Bool	RW	0	If =1 a reset/calibration sequence is performed before every B-sensor ADC input channel scan
2800 *	-	B-sensor presence mask	Bool	RW	1	Can only be 1 (B-sensor present, default) or 0 (B-sensor absent)
2900		B-sensor 64-bit identification number	Record			DS2401 or DS2405 Identification chip: unique 8-byte/64-bit number
	0	Number of entries	U8	RO	3	
	1	Lower 4 bytes	U32	RO		
	2	Upperb 4 bytes	U32	RO		
	3	Read ID and toggle DS2405 output (using 'Match ROM' command)	U32	RO		Byte 0: 0x00 if output=0, 0xFF if output=1, after toggle (*for test purposes only*)
29F0		ID-chip search	Record			Search for DS2405 ID chips (*for test purposes only*)
	0	Initialize ID-chip search	U8	RO	0	
	1	Next ID: first 4 bytes	U32	RO		
	2	Second 4 bytes of ID found	U32	RO		
	3	Next ID 'active-only': first 4 bytes	U32	RO		Finds DS2405 with output=0
	4	Second 4 bytes of ID found of 'active-only' search	U32	RO		
	5	Next ID: first 4 bytes and toggle DS2405 output	U32	RO		Byte 0: 0x00 if output=0, 0xFF if output=1, after toggle

Manufacturer-specific Profile Area (*BATCAN*) *(continued…)*						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
3000		Program Code CRC	Record			
	0	Number of entries	U8	RO	3	
	1	Check 16-bit CRC of program code in FLASH memory	U16	RO	0	SDO reply unequal to zero means there is a checksum error; absence of CRC results in SDO *Abort* with *Error Code* 1; error while accessing FLASH results in SDO *Abort* with *Error Code* 6.
	2		U16	RO	0	*not used*
	3	Get CRC	U16	RO		Return CRC from flash
3100	-	Serial Number	U32	RW		Number or 4-byte string uniquely identifying a BATCAN module, given during testing after production.
3101	-	Enable Serial Number write operation	U8	WO	**DON'T USE**	Writing 5Ah enables one write operation on the Serial Number (Object 3100).
3200		CAN-controller settings and status	Record			
	0	Number of entries	U8	RO	4	
	1	Format error interrupt counters	U32	RO		Byte 0: SERG Byte 1: CERG Byte 2: FERG Byte 3: AERG
*	2	Enable auto-start	U8	RW	0	If =1 go to *Operational* at startup
*	3	Bus-off max retry counter	U8	RW	2	Counter is decremented every 1s, but if the node reaches this maximum value it abandons re-gaining CAN-bus access
	4	Received message counter	U8	RO		Counts received CAN messages modulo 256 (for debug purposes)
3300	-	CAN Node Identifier	U8	WO		The new CAN Node Identifier is used after the next reset. (*Bootloader* firmware version 1.3 and later supports this feature, otherwise don't use it !)
3301	-	Enable CAN Node Identifier write operation	U32	WO		Writing a number that matches the Serial Number (Object 3100) enables one write operation on the CAN Node Identifier (Object 3300).

Manufacturer-specific Profile Area (*BATCAN*) *(continued…)*						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
4200		Read analogue input B-sensor	Record			24 bits analogue value
	0	Number of entries	U8	RO	7	Fixed value (see OD-index 2500, subindex 1)
	1	Input 1 (B-sensor ADC #0)	I24	RO		1st analog input:24-bit (Hall H1)
	2	Input 2 (B-sensor ADC #0)	I24	RO		2nd " " " (Hall H2)
	3	Input 3 (B-sensor ADC #0)	I24	RO		3rd " " " (Hall H3)
	4	Input 4 (B-sensor ADC #0)	I24	RO		4th " " (fullscale Hall)
	5	Input 5 (B-sensor ADC #0)	I24	RO		5th " " " (NTC)
	6	Input 6 (B-sensor ADC #0)	I24	RO		6th " " " (0ºC ref)
	7	Input 7 (B-sensor ADC #0)	I24	RO		7th " " " (100ºC ref)
4400 *	-	B-sensor NTC readings in PDO messages in degrees centigrade	Bool	RW	1	If =1 NTC ADC readings in PDO messages are converted to millidegrees centigrade (using hardcoded conversion formulas; see text)
4500 *	-	*BATsCAN* PDO compatibility	Bool	RW	0	If =1 PDO messages with B-sensor data are compatible with *BATsCAN* firmware, i.e. a data byte containing a B-sensor address or index is added (for BATCAN =0)
5C00	-	Compile-time Options	U32	RO		Bitmask denoting which compile options were used when the application code was generated (see table below for details)
5E00	-	Jump to Bootloader app	U8	WO		

Object 5C00: Compile Options		
Bit	***Option***	***Comment***
0	VARS_IN_EEPROM	Store/retrieve working copies of configuration parameters in/from EEPROM
1	–	–
2	–	–
3	–	–
4	–	–
5	AT90CAN32	Code compiled for AT90CAN32 microcontroller
6	AT90CAN64	Code compiled for AT90CAN64 microcontroller
7	AT90CAN128	Code compiled for AT90CAN128 microcontroller

Manufacturer-specific Profile Area (*BATCAN*) ***(BATsCAN compatibility area)***						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
5000		B-sensor ADC-config	Record			= Object 2500h
5100		B-sensor module status	Record			Error status, one bit per B-sensor 0=OKAY, 1=Error or Absent
	0	Number of entries	U8	RO	1	
	1	status B-sensors #0-#31	U32	RO	FFFFFFFEh	Status of 'string #0'
5200		ADC status B-sensor #0	U8	RO	0h	[1] (see footnote)
5300		ADC-reset-and-calibrate B-sensor #0	U8	WO		= Object 2600h
5380		ADC-reset-and-calibrate all B-sensors	U8	WO		= Object 2600h
5500		Read analogue input B-sensor #0	Record			= Object 4200h
5600		B-sensor address list				
	0	Total number of B-sensors	U8	RO	1	
	1	Address of 1st B-sensor	U8	RO	0	
5700		Number of B-sensors per string	Array			
	0	Total number of strings	U8	RO	4	
	1	B-sensors on string #0	U8	RO	1	
	2	B-sensors on string #1	U8	RO	0	
	3	B-sensors on string #2	U8	RO	0	
	4	B-sensors on string #3	U8	RO	0	
5800		B-sensor-to-string mapping	Array			
	0	string# with B-sensor #0	U8	RO	0	value FFh means: this B-sensor not found in any of the strings
5900		B-sensor #0 64-bit ID	Record			= Object 2900h
5B00	-	'Probe' for B-sensors	U8	RO	1	
5B03	-	Find and deselect any selected B-sensor modules	U8	RO	0	Returns the number of deselected modules
5B04	0	Select or deselect B-sensor #0 (here not using DS2405)	U8	RO	0	Returns 0 when selected, 0xFF when deselected

[1] B-sensor module ADC status byte/bitmask: **00h:** no error, **01h**: ADC reset error, **02h**: ADC calibration error, **04h**: ADC conversion time-out, **FFh**: ADC absent / not used / not in configuration.

9 Emergency Objects

CANopen *Emergency* messages are triggered by the occurrence of an internal (fatal) error situation. An *Emergency* CAN-message has the following general syntax:

BATCAN → Host

COB-ID	Byte 0-1	Byte 2	Byte 3-7
080h + *NodeID*	Emergency Error Code	Error Register (Object 1001h)	Manufacturer specific error field

A toggle bit is present in byte 7 of the Emergency message. Byte 7 alternates between the values 00h and 80h from one Emergency message to the next.

The following Emergency messages can be generated by the **BATCAN** application:

Error Description	Emergency Error Code (byte 1-0; hex)	Manufacturer-specific Error Field (byte 3-7)
CAN communication	8100	Byte 3: 00h Byte 4: total format error count Byte 5: error counter Byte 6: bus-off counter (see OD index 3200, sub 3)
CAN buffer overrun	8110h	CAN message buffer in RAM full: at least 1 message was lost
Life Guarding time-out	8130	CAN-controller has been reinitialized
RPDO: too few bytes	8210	Byte 3: minimum DLC (Data Length Code)
CRC error	5000	Byte 3: 30h Byte 4: 1 (program FLASH)
EEPROM: write error	5000	Byte 3: 41h Byte 4: Parameter block index [1] Byte 5: 0 : writing block info > 0: size of parameter block to write
EEPROM: read error	5000	Byte 3: 42h Byte 4: Parameter block index [1] Byte 5: Error id (1=CRC, 2=length, 4=infoblock)

…table continues on the next page…

[1] **0**: PDO communication parameters, **1**: Guarding parameters, **2**: ---, **3**: ---, **4**: B-sensor ADC configuration, **5**: ---, **6**: CAN configuration parameters, **7**: ---, **FFh**: BATCAN Serial Number.

Error Description	Emergency Error Code (byte 1-0; hex)	Manufacturer-specific Error Field (byte 3-7)
B-sensor ADC: conversion timeout	5000	Byte 3: 51h Byte 4: B-sensor number (0) Byte 5: ADC channel number (0..7)
B-sensor ADC: reset failed	5000	Byte 3: 52h Byte 4: B-sensor number (0) Byte 5: Error id [1]
B-sensor ADC: Hall-sensor calibration failed	5000	Byte 3: 53h Byte 4: B-sensor number (0)
B-sensor ADC: T-sensor calibration failed	5000	Byte 3: 54h Byte 4: B-sensor number (0)
B-sensor ADC problem(s) during initialisation (check OD 1002)	5000	Byte 3: 55h Byte 4: ADC status (see OD index 1002)
Irregular reset (Watchdog, Brown-out or JTAG)	5000	Byte 3: F0h Byte 4: microcontroller MCUSR register contents [2]
Bootloader: not present	5000	Byte 3: F1h
Bootloader is now in control [3]	5000	Byte 3: FEh Byte 4: 80h (81h in case of an AT90CAN128 micro) Byte 5: 64h (28h for AT90CAN128, 32h for AT90CAN32) Byte 6: microcontroller MCUSR register contents [2]
Bootloader cannot jump to application: invalid [3]	6000	Byte 3: FEh Byte 4: AAh Byte 5: AAh

Byte 2 of the *Emergency* message contains the value of the socalled *Error Register* (Object Dictionary index 1001h, a mandatory CANopen object). One or more bits of the 8-bit Error Register can be set to 1, depending on the node's history of errors since the last reset. The table below gives a description of the different bits.

Error Register (Object 1001h) bits	
Bit	**Error type**
0	generic
1	current
2	voltage
3	temperature
4	communication
5	device profile specific
6	*reserved (=0)*
7	manufacturer specific

[1] **01h**: Reset-Valid bit not set, **02h**: Reset-Valid bit not reset, **04h**: error in initial Offset Register value, **08h**: error in initial Gain Register value.

[2] AT90CANxx *MCUSR* register bits: **01h**: Power-On Reset, **02h**: External Reset, **04h**: Brown-Out Reset, **08h**: Watchdog Reset, **10h**: JTAG Reset.

[3] This Emergency message is generated by the Bootloader program !

References

[1] H.Boterenbrood
B-sensor Module, operation, readout and data-acquisition,
presentation at ATLAS Magnetic Field Workshop, 31 March 2005 (slightly outdated..).
http://www.nikhef.nl/pub/departments/ct/po/html/Bsensor/Bsensor-BfieldWS-31MAR05.pdf

[2] H.Boterenbrood,
MDT-DCS CANopen Module,
Version 2.7, NIKHEF, Amsterdam, 19 August 2011.
http://www.nikhef.nl/pub/departments/ct/po/html/MDT/MDT-DCS-CANnode.pdf

[3] F.Bergsma, H.Boterenbrood,
BsCAN3, a modular 3D B-field sensor system with CANopen read-out,
NIKHEF/CERN, Geneva, 13 May 2012.
http://www.nikhef.nl/pub/departments/ct/po/html/Bsensor/BsCAN3.pdf

[4] H.Boterenbrood,
mBATCAN, module for multiple B-sensor read-out by CAN bus and CANopen,
Version 1.1, NIKHEF/CERN, Geneva, 13 July 2012.
http://www.nikhef.nl/pub/departments/ct/po/html/Bsensor/mBATCAN.pdf

[5] H.Boterenbrood,
CANopen, high-level protocol for CAN-bus,
Version 3.0, NIKHEF, Amsterdam, 20 March 2000.
http://www.nikhef.nl/pub/departments/ct/po/doc/CANopen30.pdf

[6] CAN-in-Automation e.V.,
CANopen, Application Layer and Communication Profile,
CiA DS-301, Version 4.0, 16 June 1999.

[7] H.Boterenbrood,
CANopen Bootloader for the ELMB ATmega128 microcontroller,
Version 1.1, NIKHEF, Amsterdam, 10 March 2004.
http://www.nikhef.nl/pub/departments/ct/po/html/ELMB128/ELMBbl-doc.pdf

Appendix A. NTC Temperature Sensor Data

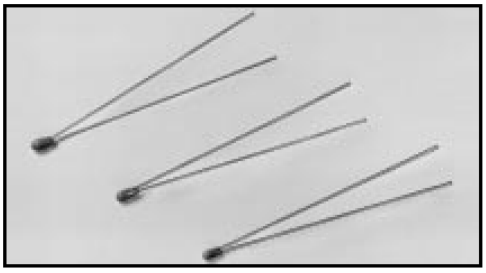
(datasheets taken from manufacturer website: http://www.thermometrics.com/)

NTC THERMISTORS: TYPE DC95

INTERCHANGEABLE CHIP THERMISTOR

DESCRIPTION:

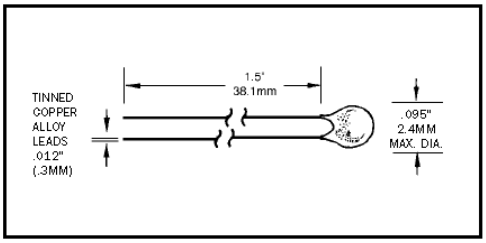
Epoxy coated interchangeable chip thermistors with bare tinned copper lead-wires.



FEATURES:

- Precision, solid state temperature sensor
- Interchangeability down to ±0.1 °C
- Suitable for use over the range of –80°C to 150°C
- High sensitivity greater than –4%/°C at 25°C
- Suitable for temperature measurement, control and compensation
- High reliability and stability over interchangeable range
- Most popular R-vs-T curves are available
- Resin coated for good mechanical strength and resistance to solvents
- .012" (.3 mm) dia. bare tinned copper lead-wires

DIMENSIONS:



Select appropriate part number below for resistance and temperature tolerance desired

$R_{25°C}$	MATERIAL SYSTEM	± .1°C 0°C to 70°C	± .2°C 0°C to 70°C	± .2°C 0°C to 100°C
2000	F	DC95F202V	DC95F202W	DC95F202Z
2252	F	DC95F232V	DC95F232W	DC95F232Z
3000	F	DC95F302V	DC95F302W	DC95F302Z
5000	F	DC95F502V	DC95F502W	DC95F502Z
10000	F	DC95F103V	DC95F103W	DC95F103Z
10000	Y	DC95Y103V	DC95Y103W	DC95Y103Z
30000	H	DC95H303V	DC95H303W	DC95H303Z
50000	G	DC95G503V	DC95G503W	DC95G503Z
100000	Y	DC95Y104V	DC95Y104W	DC95Y104Z
100000	G	DC95G104V	DC95G104W	DC95G104Z

OPTIONS:

Consult factory for availability of options:

- Other resistance values in the range of 100Ω - 100kΩ
- Other tolerances or ranges
- Alternative lead-wires or lengths
- Non standard R-vs-T curves
- Controlled dimensions

DATA:

THERMAL AND ELECTRICAL PROPERTIES:

Dissipation constant:.....(still air) 1 mW/°C
(stirred oil) 8 mW/°C

Thermal time constant:.....(still air) 10 sec.
(stirred oil) 1 sec.

Maximum power at 25°C 75mW
(derated from 100% at 25°C to 0% at 100°C)

BOWTHORPE THERMOMETRICS
Crown Industrial Estate, Priorswood Road
Taunton, Somerset TA2 8QY UK
Tel +44 (0) 1823 335200
Fax +44 (0) 1823 332637

THERMOMETRICS, INC.
808 US Highway 1
Edison, New Jersey 08817-4695 USA
Tel +1 (732) 287 2870
Fax +1 (732) 287 8847

KEYSTONE THERMOMETRICS CORPORATION
967 Windfall Road
St. Marys, Pennsylvania 15857-3397 USA
Tel +1 (814) 834 9140
Fax +1 (814) 781 7969

MATERIAL TYPE: F

AVAILABLE PRODUCTS:

HM, C100, EC95, DC95, MC65, MF65, SC30, SC50

Data for material type : F

Temp Range (°C)	Ratio	Beta
0 to 50	9.08	3895
0 to 70	18.64	3917
25 to 50	2.78	3933
25 to 85	9.30	3969
25 to 100	14.64	3981
25 to 125	29.05	3999
37.8 to 104.4	9.67	4000

To calculate Rt/R25 at temperatures other than those listed in the table, use the following equation:

$Rt/R25 = \exp\{A + B/T + C/T^2 + D/T^3\}$

where T = temperature in K

where K = °C + 273.15

Temp Range (°C)	A	B	C	D
-50 to 0	-1.4122478E+01	4.4136033E+03	-2.9034189E+04	-9.3875035E+06
0 to 50	-1.4141963E+01	4.4307830E+03	-3.4078983E+04	-8.8941929E+06
50 to 100	-1.4202172E+01	4.4975256E+03	-5.8421357E+04	-5.9658796E+06
100 to 150	-1.6154078E+01	6.8483992E+03	-1.0004049E+06	1.1961431E+08

To calculate the actual thermistor temperature as a function of the thermistor resistance, use the following equation:

$1/T = a + b(\ln Rt/R25) + c(\ln Rt/R25)^2 + d(\ln Rt/R25)^3$

Rt/R25 range	a	b	c	d
68.600 to 3.274	3.3538646E-03	2.5654090E-04	1.9243889E-06	1.0969244E-07
3.274 to 0.36036	3.3540154E-03	2.5627725E-04	2.0829210E-06	7.3003206E-08
0.36036 to 0.06831	3.3539264E-03	2.5609446E-04	1.9621987E-06	4.6045930E-08
0.06831 to 0.01872	3.3368620E-03	2.4057263E-04	-2.6687093E-06	-4.0719355E-07

†The deviation resulting from the tolerance on the material constant, Beta. The deviation must be added to the resistance tolerance of the part as specified at 25°C.

Temperature (°C)	Rt/R25 nominal	Temp Coef ∝ (%/°C)	β Deviation† (±%)
-50	68.60	7.21%	2.30%
-45	48.16	6.96%	2.68%
-40	34.23	6.71%	2.87%
-35	24.62	6.48%	2.92%
-30	17.91	6.26%	2.86%
-25	13.17	6.05%	2.71%
-20	9.782	5.85%	2.50%
-15	7.339	5.66%	2.25%
-10	5.558	5.47%	1.97%
-5	4.247	5.30%	1.68%
0	3.274	5.13%	1.37%
5	2.544	4.97%	1.07%
10	1.992	4.81%	0.78%
15	1.572	4.67%	0.50%
20	1.250	4.53%	0.24%
25	1.000	4.39%	0.00%
30	0.8056	4.26%	0.21%
35	0.6530	4.14%	0.40%
40	0.5326	4.02%	0.56%
45	0.4369	3.91%	0.69%
50	0.3604	3.80%	0.80%
55	0.2989	3.69%	0.87%
60	0.2491	3.59%	0.92%
65	0.2087	3.49%	0.93%
70	0.1756	3.40%	0.92%
75	0.1485	3.31%	0.88%
80	0.1261	3.23%	0.81%
85	0.1075	3.14%	0.72%
90	0.09209	3.06%	0.59%
95	0.07916	2.99%	0.45%
100	0.06831	2.91%	0.28%
105	0.05916	2.85%	0.08%
110	0.05141	2.77%	0.12%
115	0.04483	2.70%	0.36%
120	0.03922	2.64%	0.61%
125	0.03442	2.57%	0.87%
130	0.03030	2.51%	1.16%
135	0.02675	2.47%	1.46%
140	0.02369	2.41%	1.82%
145	0.02103	2.35%	2.14%
150	0.01872	2.35%	2.46%

BOWTHORPE THERMOMETRICS
Crown Industrial Estate, Priorswood Road
Taunton, Somerset TA2 8QY UK
Tel +44 (0) 1823 335200
Fax +44 (0) 1823 332637

THERMOMETRICS, INC.
808 US Highway 1
Edison, New Jersey 08817-4695 USA
Tel +1 (732) 287 2870
Fax +1 (732) 287 8847

KEYSTONE THERMOMETRICS CORPORATION
967 Windfall Road
St. Marys, Pennsylvania 15857-3397 USA
Tel +1 (814) 834 9140
Fax +1 (814) 781 7969